A Dual Network Solution (DNS) for Lag-Free Time Series Forecasting

Subhrajit Samanta
ERI@N-IGS
NTU
Singapore
sama0021@e.ntu.edu.sg

Mahardhika Pratama
SCSE
NTU
Singapore
MPRATAMA@ntu.edu.sg

Suresh Sundaram
Aerospace Engineering
IISC
Bengaluru, India
sureshsundaram@iisc.ac.in

Narasimalu Srikanth
ERI@N
NTU
Singapore
nsrikanth@ntu.edu.sg

***Abstract*—When it comes to time series forecasting, lag in the predicted sequence can be a predominant issue. Unfortunately, this is often overlooked in most of the time series literature as this does not contribute to a high prediction error (i.e. MSE). However, it leads to a rather poor forecast in terms of movement prediction in time series. In this article, we tackle this basic problem with a novel trend driven mechanism. Trend, defined as the inherent pattern of the data, is extracted here and utilized next to perform a lag-free forecasting. We propose a generic and light Dual Network Solution (DNS), where the first network predicts the trend and the second network utilizes that predicted trend along with its historical information to capture the dynamical behavior of the time series efficiently. DNS exhibits a substantially improved ($\approx 10\%$ better) performance compared to more complex and resource-intensive state-of-the-art algorithms in large scale regression problems. Apart from the traditional Mean Squared Error (MSE), we also propose a new Movement Prediction Metric or MPM (for detection of lag in time series) as a new complementary performance metric to evaluate the efficacy of DNS better.**

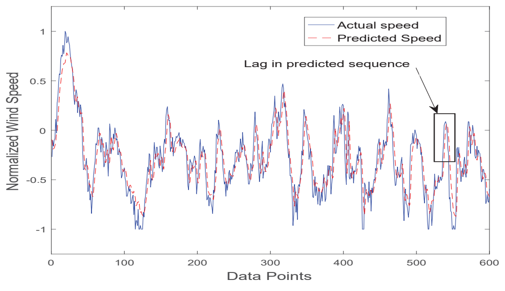
***Index Terms*—Lag in time series prediction, Dual network solution, Movement prediction, Lag free time series prediction**

I. Introduction

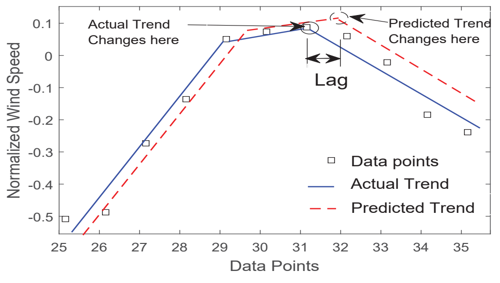
Time series modelling and forecasting are one of the most popular areas of research in the machine learning and data science community for its wide applicability in various domains across different sectors such as energy demand prediction, weather forecasting, financial prediction, etc. This ever-growing field of study has seen significant improvement over the past decades. Starting from the statistical methods of regression analysis to recent advents in machine learning algorithms have propelled this research to a new high. However, across most of this literature, a basic problem of lag difference (in the predicted sequence) persists. Next, we discuss the problem elaborately to highlight our motivation.

A. Problem Description : Lag in Time Series

A time series model utilizing historical data alone can detect a change in the trend only after the historical data also experiences the same change. Naturally, the change prediction is late which leads to the problem of lag. To explain this problem further we take an example case: wind speed [1] forecasting. The prediction task is performed here with a Multi-Layer Perceptron network or MLP (hidden node with sigmoid activations in the single hidden layer) with only past values of the target sequence as inputs and the actual vs prediction plot is provided in figure 1. From time instance 25 to 31 in figure 1b the trend is upward which leads the MLP to predict an upward trend at point 31 (marked in the figure 1b) in spite of an actual downward trend. This results in a predicted sequence with a distinct lag as seen in the figure 1a and 1b.



(a) Actual vs prediction plot



(b) Zoomed in to show Lag

Fig. 1: Demonstration of Lag in Time Series Forecasting

The lag in the predicted sequence results in a 'close-by, past value' prediction hence it does not contribute to a high prediction error such as MSE. Besides, MSE does not take into account the movement direction of the prediction (as the squared error ignores its sign), therefore it is unable to detect lag. For instance, the wind forecast in figure 1a looks like a good forecast however suffers from the lag problem. Sometimes due to over-fitting of the historical values, the result can be precisely a curve that mimics the real values almost

[1] http://mesonet.agron.iastate.edu/request/awos/1min.php

perfectly, with just a little delay i.e. 'some time late'. For further confirmation, it is sufficient to observe the curve in the proximity of the local maximums (i.e near point 31 in figure 1b). If we take another example of stock price prediction, a lag of this nature can lead the forecast to predict an upwards movement when the actual movement is already downwards, thus possibly leading to a huge financial loss thereby further affirming the significance of lag problem. This issue is very common throughout most time series literature and to the best of our knowledge has not been addressed properly yet as discussed in the related work. Next, we present a brief review of the relevant literature in the field of time series forecasting (both classical and machine learning based methods).

B. Related Work

Time series forecasting is one of the most highly researched areas for its widespread applications across varied domains. Before the advent of machine learning, classical statistical methods such as Auto-Regressive Moving Average (ARMA) [1] were most popular. However, generally they assume linear functional relationship for modeling the dynamic behavior which is not always highly effective. Recently time series research has immensely benefited from the advancements in the area of recurrent neural networks.

Recurrent neural networks were initially proposed by [2]. Exploding and vanishing gradients were one of the major problems of training RNNs with backpropagation [3] until Long Short-Term Memory [4] was proposed. LSTM learns both long and short term dependencies using gate activations. A modification on LSTM referred to as the Gated Recurrent Unit (GRU) [5] was recently proposed where input and forget gates of LSTM are combined into an update gate. There have been other researches which build on top of the basic LSTM including Bidirectional RNN [6] which employs both past and future information, Phased LSTM [7] where hidden states are updated asynchronously, State Frequency Memory Network [8] or SFM which separates dynamical patterns in the frequency domain and offers a detailed analysis of temporal sequences. However, these models perform poorly sometimes in short-term time series problems [9].

There has been a plethora of researches in the area of so-called shallow learning which addresses the problem of small scale short-term time series forecasting more suitably. Fuzzy-neural networks are popular for their uncertainty handling capabilities and interpretability. Some of the recent works in this field include [10], [11], [12], [13], [14], [15] etc. Recently attention mechanism has seen a growing interest and the same is adopted in the recurrent neural network as well i.e. in [16], [17] local time-frequency features are captured along with global long-term trend and fused to perform superior forecasting.

Real-world application-wise, prediction of a time series movement is sometimes more beneficial hence desirable than the prediction of actual values alone, as shown in [18], [19]. Most time series algorithms (as discussed above) generally utilize historical data alone for the modeling purpose, making them prone to the problem associated with lag i.e. poor movement prediction. Few statistical works which focus on trend extraction and prediction are [20], [21] . To the best of our knowledge, only deep learning method directly focusing on predicting trend is TreNet [22] where a hybrid of LSTM and CNN is utilized. However this work does not provide an absolute value forecast and concentrates solely on estimating the trend, thus the issue of lag in the predicted sequence still remains unexplored. Therefore, in this paper, a trend-driven dual network solution or DNS is proposed to address the issue of lag and the major contributions are,

- **Trend Extraction & Utilization :** Time series datasets are often riddled with fluctuations stemming from noise and seasonality. Hence, learning with historical data alone often leads to overfitting causing lag (as discussed before). To address this, first we extract the trend with two proposed methodologies i.e. Moving Average Method and Slope Difference Method. Then we utilize the extracted trend as the target sequence (instead of the actual values) in a supervised manner and learn to predict the trend. The predicted trend drives the final forecast.
- **Dual Network Solution :** A novel Dual Network Solution (DNS) is proposed here. The first network (Trend Prediction Network) learns to map the exogenous input features to the extracted trend. The predicted trend (from the first network) along with its finite past instances are passed on to the second (Temporal) network to learn the intricate details of the system dynamics. Therefore, we have a trend driven temporal network which is able to provide an improved forecast (by $\approx 10\%$ than its peers) where the predicted curve follows the actual curve closely without lag. An overview of DNS is provided in figure 2.
- **New Performance Metric :** Most of the error based performance metrics (such as MSE) often fail to detect the presence of lag in a time series forecast. Hence, we propose a new Movement Prediction Metric (MPM) as a complementary measurement to evaluate time series models better. DNS shows both low MSE and high MPM.

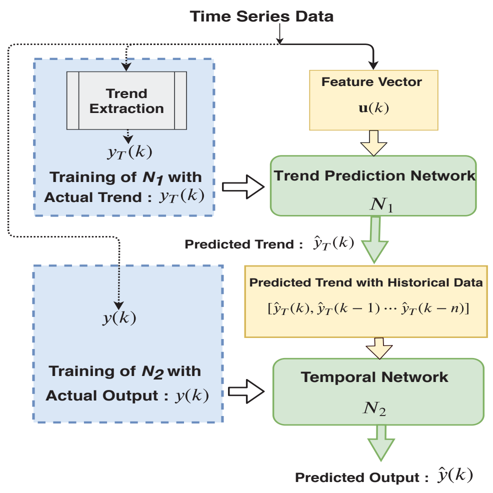

Time Series Data
Trend Extraction
$y_T(k)$
Training of N_1 with Actual Trend : $y_T(k)$
Feature Vector $\mathbf{u}(k)$
Trend Prediction Network N_1
Predicted Trend : $\hat{y}_T(k)$
Predicted Trend with Historical Data $[\hat{y}_T(k), \hat{y}_T(k-1) \cdots \hat{y}_T(k-n)]$
$y(k)$
Training of N_2 with Actual Output : $y(k)$
Temporal Network N_2
Predicted Output : $\hat{y}(k)$

Fig. 2: Overview of the Dual Network Solution (DNS)

II. Trend Extraction

We need to extract the trend to train the first network. Two separate methods can be employed for this purpose,

A. Moving Average Method (MAM)

We employ the principle of moving average on the time series sequence to get the trend. First, we set a window of size $W+1$. For each data point in the sequence, we consider the W neighboring samples centering it and get their mean. This process is done for all the data points in the sequence i.e. $y(k)$ and the trend $y_T(k)$ is,

$$y_T(k) = \frac{1}{W+1} \sum_{i=-W/2}^{W/2} y(k+i) \tag{1}$$

The average over the neighboring samples helps in smoothing out the sequence hence we are able to get rid of noise and frequent fluctuations from the time series and extract the trend as shown figure 3. The bigger the window size the smoother is the trend curve. For example, to extract the trend in figure 3 we have utilized a window size of 21 (i.e. $W = 20$).

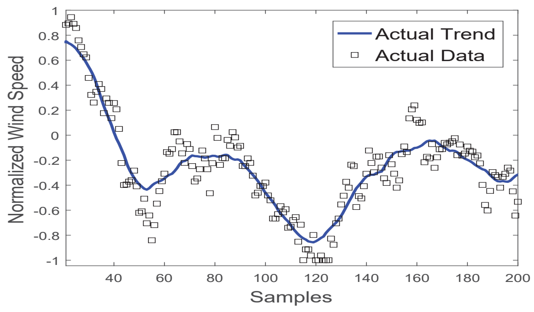


Fig. 3: Trend in wind dataset extracted with MAM

Instead of mean, a median should be utilized to extract the trend if the time series data is prone to outliers.

B. Slope Difference Method (SDM)

The moving average method is offline in nature and requires all the training samples to be present. Hence it is not suitable in an online context hence the second method is proposed.

First, we compute the slope of the current trend. Starting with $y(k)$ and $y(k+1)$ we represent the trend at k^{th} instance with slope $m_T(k) = \frac{y(k+1)-y(k)}{(k+1)-k} = y(k+1) - y(k)$. Let us imagine a new $(k+i)^{th}$ sample arrives and makes a slope of $m(k+i)$ with the k^{th} sample. If the new slope is significantly different than $m_T(k)$ then we will conclude a change in trend and reassign the latest slope made by $(k+i)^{th}$ sample as the current trend and so on. To summarize,

$$\begin{aligned} m_T(k+i) &= m(k+i) \quad \text{if} \quad |m(k+i) - m_T(k)| \geq \epsilon \\ &= m_T(k) \quad \text{if} \quad |m(k+i) - m_T(k)| < \epsilon \end{aligned} \tag{2}$$

with ϵ as the threshold. From the slope defined in equation 2 the trend can be retrieved easily using,

$$\begin{aligned} y_T(k+i) &= m_T(k).\big((k+i) - i\big) + y(k) \\ &= m_T(k).i + y(k) \end{aligned} \tag{3}$$

Figure 4 demonstrates an example of the extracted trend for the same dataset using the slope difference method with $\epsilon = 0.01$. Next, we discuss in detail the proposed dual network solution (DNS) as briefed in the outline.

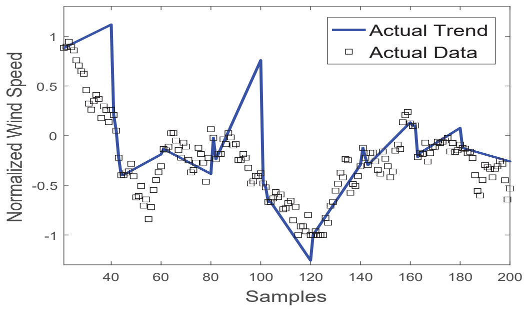


Fig. 4: Trend in wind dataset extracted with SDM

III. Trend Prediction Network (N_1)

The first network denoted with N_1 is accountable for mapping the input features (factors) to the extracted trend (as shown in the last section). Let us denote the feature vector as $\mathbf{u}(k) \in \mathbb{R}^P$ where the number of input features are P and k is the time instance. From previous section we already have the trend associated with the data points i.e. $y_T(k)$. Unlike regular time series model here we utilize the extracted trend as the target sequence (i.e. $y_T(k)$ instead of $y(k)$) to train our network N_1 (which is feed-forward in principle). Thus the network learns to predict the trend $\hat{y}_T(k)$ from the exogenous input vector,

$$\hat{y}_T(k) = f_1\big(\mathbf{u}(k)\big) \tag{4}$$

Remark 1 : One of the major advantages of DNS lies in the fact that any kind of neural network can be employed to build the N_1 platform (making DNS generic) as the primary goal here is to approximate the function $f_1(.)$ in equation 4.

A. Network Structure and Update

For the experiments conducted in this study we have utilized a radial basis function [23] network as N_1, primarily for its ease of implementation along with its well suitability in time series modeling. Hence, N_1 here is a 3 layer network, with P input nodes, R hidden nodes and 1 output node which provides weighted ($\mathbf{w} \in \mathbb{R}^R$) summation of the Gaussian activations $\mathbf{F} \in \mathbb{R}^R$ such that $\hat{y}_T(k) = \mathbf{w}^T\mathbf{F}$. To train the network, a new mini-batch variant of projection based learning or PBL [24] method is proposed. First, we divide the training dataset into smaller batches and compute the prediction errors. The sum of the squared errors over a mini-batch (of size c) is used as the cost function here i.e. $J_1 = \frac{1}{2}\sum_c \big(y_T(k) - \hat{y}_T(k)\big)^2 = \frac{1}{2}\sum_c \big(y_T(k) - \mathbf{w}^T\mathbf{F}\big)^2$. Optimal weight $\mathbf{w}^\star$ is obtained when the cost function is minimized or $\frac{dJ_1(\mathbf{w})}{d\mathbf{w}}\Big|_{\mathbf{w}=\mathbf{w}^\star} = 0$. Rearranging we get,

$$\mathbf{w}^\star = A^{-1}B \tag{5}$$

where, $A \in \mathbb{R}^{RXR}$ consists of a_{rr^*} and $B \in \mathbb{R}^{RX1}$ of b_r s.t.

$$a_{rr^*} = \sum_c F_r.F_{r^*} \quad , \quad b_r = \sum_c F_r.y_T \tag{6}$$

for $r, r^* = 1, 2, \cdots, R$. N_1 learns from these this mini-batches over multiple epochs to reach the optimal weights.

Remark 2 : A regular mini-batch gradient based backpropagation can also be adopted to train N_1. However, the proposed mini-batch PBL method is adopted here, as it does not suffer from the problems of exploding or vanishing gradients.

IV. Temporal Network (N_2)

Generally historical values of the actual sequence are used to model a time series, however using them alone often leads to a 'close by past value' prediction with lag. Hence in the second network (N_2) of the DNS, we utilize the predicted trend instead, to obtain a lag-free forecast. N_2 is temporal in nature to capture the system dynamics deftly. It utilizes the predicted trend $\hat{y}_T(k)$ from N_1 along with its past instances (i.e. $\hat{y}^T(k-1), \cdots \hat{y}^T(k-n)$) to perform the final prediction,

$$\hat{y}(k) = f_2\big(\hat{y}_T(k), \hat{y}_T(k-1), \hat{y}_T(k-2) \cdots \hat{y}_T(k-n)\big)\big) \tag{7}$$

here $f_2(.)$ is the nonlinear function representing the prediction task and n is the number of past instances required in N_2 to perform this function approximation. Thereby, N_2 is also independent of the underlying neural network platform.

Remark 3 : We feed the predicted trend $\hat{y}_T(k)$ to N_2 instead of the actual trend $y_T(k)$ because, during testing (or application), the actual trend will not be available yet.

To design the temporal network, we need to determine the required number of past instances n (from equation 7). A Bayesian method of Past Dependency Estimation or PDE inspired from recently published LEMON method [25], is adopted here for this purpose. A quick overview of the same is provided in figure 5. PDE should be considered as a preprocessing step before N_2 actually starts to learn.

A. Past Dependency Estimation (PDE)

To estimate the past dependency, first we train the network N_2 without any recurrence (i.e. predict $\hat{y}(k)$ with only $\mathbf{v}(k) = \hat{y}_T(k)$ as input) like a feed-forward network to provide it with some initial knowledge using the first mini-batch of the data. Once the network acquires a preliminary knowledge, the PDE starts. Past dependency is computed from the next mini-batch.

Beginning with an initial memory of size M, N_2 performs its prediction task for each sample. Past prediction errors are retained in the memory in a first-in-first-out manner so at k^{th} time instance the memory window will contain $e^M(k) = [e_0(k), e_1(k) \cdots e_{M-1}(k)]^T$ where $e_j(k) = y(k) - \hat{y}(k-j)$ is the prediction error if $(k-j)^{th}$ sample was employed alone to perform the k^{th} prediction task (for $j = 0, 1 \cdots (M-1)$).

The error vector $e^M(k)$ is utilized to compute the likelihood of a correct prediction. Assuming a normal distribution for the prediction errors (confirmed by Lilliefors test) with zero mean (as the ideal prediction error should be zero) and the standard deviation σ set from the variance in the error vector $e^M(k)$, the likelihood of correct prediction of the k^{th} sample output $y(k)$ utilizing only $(k-j)^{th}$ input $v(k-j)$ i.e. $P(y(k)|\mathbf{v}(k-j)$ is obtained from $\mathcal{N}(0, \sigma^2)$.

As the prior distribution is unknown, we start with a flat (uniform) prior i.e. $P(\mathbf{v}(k-j)) = 1/M$, compute the posteriors $P(\mathbf{v}(k-j)|y(k))$, and then update the priors with resultant posteriors in a weighted recursive manner i.e.

$$P(\mathbf{v}(k+1-j)) = \alpha P(\mathbf{v}(k-j)) + (1-\alpha) P(\mathbf{v}(k-j)|y(k)) \tag{8}$$

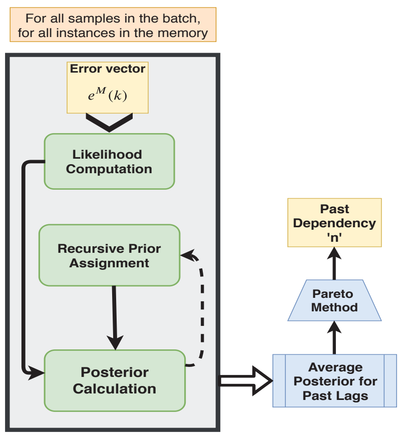
The posterior probability of $\mathbf{v}(k-j)$ i.e. $P(\mathbf{v}(k-j)|y(k))$ represents the importance of $\mathbf{v}(k-j)$ in prediction of the k^{th} instance and the same is derived using Bayes' formula,

$$P(\mathbf{v}(k-j)|y(k)) = \frac{P(y(k)|\mathbf{v}(k-j)).P(\mathbf{v}(k-j))}{\sum_{j=0}^{M-1}[P(y(k)|\mathbf{v}(k-j)).P(\mathbf{v}(k-j))]} \tag{9}$$

Then the posteriors of each past instances in the memory, are averaged over all the samples in the second mini-batch, $PP_{avg}(j) = \sum_c P(\mathbf{v}(k-j)|y(k))/c$. A Pareto based approach is utilized next to extract the past dependency measure from the average posteriors. The Pareto principle or the 80-20 rule states that the often 80% effects stem from 20% factors. Hence, the past instances in the memory window constituting total 80% of the posterior probability (i.e. 0.8; as the sum of all the posteriors is 1) will represent the past dependency,

IF $\sum_{j=0}^{i} PP_{avg}(j) \leq 0.8$ **THEN** $n = i$

Once the past dependency is estimated we can conclude that N_2 requires n number of past instances of the predicted trend to approximate the function $f_2(.)$ for the final forecast.



For all samples in the batch, for all instances in the memory
Error vector
$e^M(k)$
Likelihood Computation
Recursive Prior Assignment
Posterior Calculation
Average Posterior for Past Lags
Pareto Method
Past Dependency 'n'

Fig. 5: Overview of Past Dependency Estimation Process

B. Network Structure and Update

Next, we feed a second RBF network with the predicted trend along with n number of its past instances to build N_2. It learns the inherent dynamics of the time series from the historical data using the mini-batch PBL method with cost function $J_2 = 1/2 \sum_c \big(y(k) - \hat{y}(k)\big)^2$. Thereby it is able to provide an improved lag-free forecast driven by the trend.

Remark 4 : Traditionally the problem of past dependency (i.e. setting the hyperparameter value of n in equation 7) is addressed in an empirical manner hence we have employed the pre-processing step of PDE to reduce handcrafting. The n we obtain from PDE also helps in reducing over or under-fitting.

V. Experimental Setup

In this study, an RBF network with 100 hidden nodes is adopted to build the first network N_1. A second RBF network (also with 100 hidden nodes) is used to build N_2. PDE process (with $M = 10$ and $\alpha = 0.9$) estimates the number of past instances (of predicted trend) required i.e. n to design N_2 from the first two mini-batches (first one for preliminary knowledge, next one for the PDE). A mini-batch size of ($c = 50$) is used to train both N_1 and N_2 with the mini-batch PBL method over 100 epochs. We explore the effect of both MAM (with varying window size $W + 1$) and SDM (with varying threshold ϵ) on the final forecast. For the actual vs predicted curves (in figure 6,7,8) and performance comparison in table 5, we pick the best performing ones between MAM and SDM.

A. Discussion on Hyperparameter Setting

The initial memory window M is the maximum number of past samples to account during PDE hence the estimated dependency will always be less than M i.e. $n \leq M$. A smaller or bigger memory window can lead to different measurements of n (because of the distribution of the priors and posteriors in the memory changes with its width). Here $M = 10$ provided the best accuracy. The priors are updated with last posteriors as shown in equation 8 and for that a prior regulating factor α of 0.9 is used. A higher α is employed so that the prior values do not turn too alike with the posteriors only after a few samples. A Pareto inspired approach is adopted here to estimate the past dependency where past instances accounting for 80% of the posteriors are regarded as the measurement of past dependency. A threshold of less or more than 80% can also be utilized for the same however they might lead to under or over-estimation.

B. Performance Metrics

MSE : As per standard practice in time series literature, we utilize Mean Squared Error or MSE to compare the performance of DNS with other state-of-the-art methods during testing. Mean squared error is defined as the average of the squared prediction errors,

$$MSE = \frac{1}{S} \sum_{S} \big(y(k) - \hat{y}(k)\big)^2 \tag{10}$$

with S as the number of samples. However, MSE cannot be the sole indicator of performance as it often fails to trace lag (as discussed in section 1.1). Hence, in addition we propose a Movement Prediction Metric (MPM) as complementary.

Proposed MPM: MSE does not consider the direction of prediction as it squares the error. Hence, we propose an indicator which is not error based rather it looks only at the prediction movement. If $y(k+1) \geq y(k)$ then it is considered as a real upward movement and vice versa. If DNS prediction : $\hat{y}(k+1) \geq \hat{y}(k)$ then it is counted as a True Up prediction or TU. Similarly True Down (TD), False Up (FU) and False Down (FD) predictions can be counted and we can define the Movement Prediction Metric (MPM) as,

$$MPM = \frac{TU + TD}{TU + FU + TD + FD} \tag{11}$$

If there is lag present then the predicted curve will fail to track the actual curve suitably in the presence of optimas, hence it will lead to an overall low MPM value. For a highly overfitted prediction mimicking the historical data as in figure 1a, the MPM value hovers around 50%. Higher MPM value indicates to an improved forecast without lag.

VI. Experimental Results

First, we utilize the aforementioned wind dataset to demonstrate the improved lag-free forecast with DNS. The efficacy of MAM and SDM on the final prediction performance is explored here. In the second category, we employ two real-world large scale regression problems namely stock price and power consumption prediction to show the superior performance achieved with our method compared to other state-of-the-art time series algorithms.

A. Demonstration of lag-free forecast

We collect wind speed data along with the directional information for February 1, 2011, to February 28, 2011, containing 600 samples from the Iowa (USA) Department of Transport website. More details can be found in [26]. We have used the directional information as the feature $u(k)$ to map the extracted trend in N_1. PDE estimates $n = 4$.

Moving Average Method (MAM) : Different sizes of windows (from small to large) are employed to extract the trend and utilized in N_1. Predicted trend along with its historical data are utilized in N_2. The corresponding MSE and MPM are recorded in table I.

TABLE I: Performance with Moving Average Method

Window size ($W+1$)	MSE	MPM (in %)
3	0.0064	74.10
5	0.0077	**74.46**
11	0.0086	74.10
21	0.0164	69.46

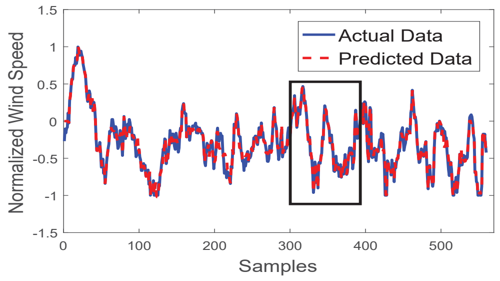
We see that a larger window size results in higher MSE, however the decrease in the movement prediction is not very drastic. A window size of 5 produced the best MPM $\approx 75\%$, indicating an improved lag-free forecast.

Slope Difference Method (SDM) : Similarly we utilize the slope difference method with varying thresholds to evaluate its efficacy on the forecasting performance of the proposed dual network system. The results are provided in table II.

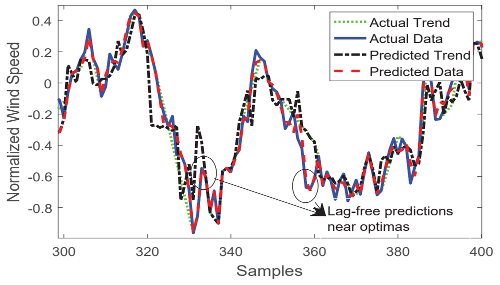
TABLE II: Performance with Slope Difference Method

Difference Threshold (ϵ)	MSE	MPM (in %)
0.001	0.0059	**78.03**
0.005	0.0061	77.85
0.010	0.0064	77.67
0.050	0.0077	74.43

The extracted trend from SDM is not as smooth as the trend extracted from MAM. The trend here follows the actual curve more closely than MAM when the threshold is low. For the wind problem, SDM provided a better MSE and MPM as shown in table II. However with SDM, the result deteriorates faster with higher value of the threshold ϵ as with high threshold value, a lot of slope change information gets ignored (from equation 2-3). In figure 6a we provide the actual vs predicted plot for the best case from the use cases (i.e. SDM with $\epsilon = 0.001$) whereas figure 6b shows the lag-free improved forecast achieved for the same in a zoomed format.



(a) Performance of SDM with $\epsilon = 0.001$



(b) Zoomed in to show lag-free forecast

Fig. 6: Wind Speed Prediction with DNS

B. Performance Comparison

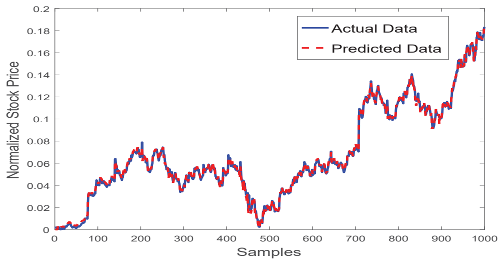
In this section, we employ two different large scale time series problems to show the superiority of the proposed DNS compared to other state-of-the-art regression models.

1) Stock Price Prediction: The stock problem deals with the prediction of daily opening stock prices with the previous day's information. We collect the stock data of 50 different organizations (from ten different sectors) for a period of 2007-2016. The dataset is downloaded directly from the Github repository [2]. We utilize 2007-2014 data for training, 2015 data for validation and 2016 data for testing for each of these 50 cases. More details on the data are available in [27]. During testing, test MSE is computed for each of these fifty stock time series and their overall average is reported as the performance index in table V.

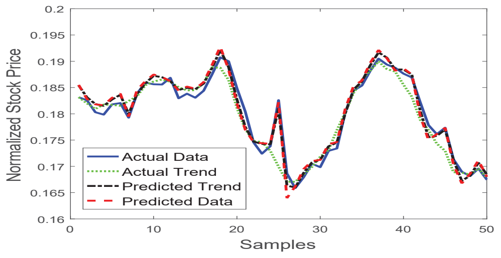
To train the first network we have utilized previous day's high price, low price, close price and volume of sale as the input features $\mathbf{u}(k)$ to map the trend (of next day's opening). 3 past instances are utilized in N_2 (from PDE) for the final prediction. We pick one stock (Amazon) out of the 50 to present the efficacy of the trend extraction methods in terms of MSE (non-normalized) and movement prediction in table III. A window size of 3 produced the best movement prediction for Amazon stock price prediction. Sections of these actual vs. predicted plots are provided in figure 7a and 7b.

TABLE III: Single step Amazon Stock Price Prediction

MAM Window $(W+1)$	Test MSE	Test MPM in%
3	**3.512**	**77.00**
5	3.636	76.00
SDM Threshold (ϵ)	**Test MSE**	**Test MPM in %**
0.001	6.813	70.00
0.003	11.213	58.00



(a) Training plots for Amazon stock



(b) Testing plots for Amazon stock

Fig. 7: Amazon Stock Price Prediction with DNS

2) Power Consumption Prediction: The power consumption pattern in a household in France is predicted in this forecasting problem. Part of the data is downloaded from UCI-ML repository[3] and it contains the details of electric power consumption in one household for the year of 2010. The voltage sequence is the target time series as per the standard practice. More details on the problem and train-validation-test partition are in [16].

We utilize sub-meter readings and active power as external features to map the trend. PDE estimates $n = 5$ for N_2. Performance of MAM and SDM on this single step ahead prediction are provided in table IV. SDM with $\epsilon = 0.001$ provides the best result here and the corresponding actual vs predicted curves (for a section of the data) are provided in fig. 8a and 8b. Performance comparison is in table V.

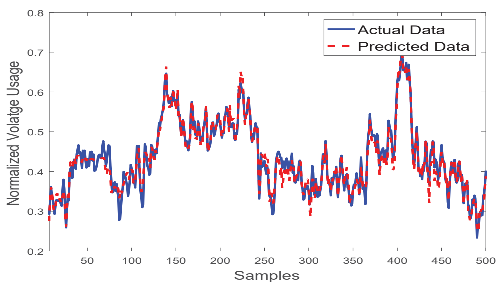
Remark 5 : Stock price data often contains a higher amount of volatility (thus frequent fluctuations) compared to the power

[2]https://github.com/z331565360/State-Frequency-Memory-stock-prediction

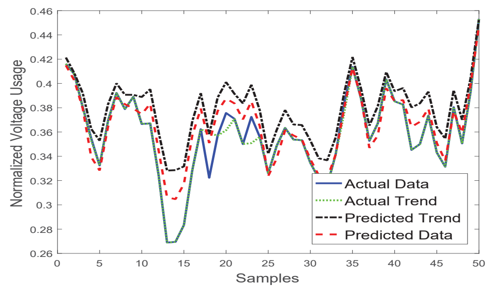
[3]https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption

TABLE IV: Single step Power Consumption Prediction

MAM Window $(W+1)$	Test MSE	Test MPM in %)
3	0.223	92
5	0.461	90
SDM Threshold (ϵ)	**Test MSE**	**Test MPM in %)**
0.001	**0.272**	**94**
0.010	0.354	92



(a) Training plots for Power consumption



(b) Testing plots for Power consumption

Fig. 8: Power Consumption Prediction with DNS

consumption data. Hence MAM is more suitable here to extract the trend. The past dependency measure is also lesser for the stock problem, signifying a low reliance on long past which is apt considering the swift changing nature of the stock market.

3) Baseline Methods for Comparison: For the purpose of performance comparison, we resort to MSE as per the standard practise in time series literature. We provide test MSE for short term (1 step ahead), mid-term (3- step ahead) and long-term (5-step ahead) predictions. The baselines are described briefly,

- **Naive** approach as the name suggests considers the previous output as the prediction.
- **ARIMA** [1] is a well-known classical method of time series prediction.
- **Support Vector Regression (SVR)** is the modified version of support vector machine for regression. More details on the design is availble in [31], [28].
- **Artificial Neural Network (ANN)** are popular in the machine learning community for their data-driven learning capabilities. In this study we utilize ANN from [29].
- **Long Short Term Memory** is a variant of recurrent neural network mainly utilized for language modeling. More details on the design are available in [4].
- **State Frequency Memory (SFM)** [8] separates dynamical patterns in the frequency domain for detailed analysis.
- **Attentive Neural Network** [16] traces local time-frequency features along with the long-term pattern.
- **Convolutional Neural Network (CNN)** are widely practised in the domain of computer vision and image classification. Here it performs forecast with convolution on wavelet transformed time series i.e. scalogram ([30]).
- **Ensemble of CNN and LSTM** as the name suggests, fuses these two networks (LSTM and CNN) to perform time series forecasting as introduced by [22].
- **Radial Basis Function Network** [23] employs Gaussian activation in its hidden layer. A RBF (100 nodes) with required past instances (estimated from PDE) of target as input i.e. $y(k-1), y(k-2)\cdots y(k-n)$ is also employed.

The purpose of the last comparison is to emphasize on the performance improvement achieved when the same RBF network is utilized in the proposed dual network framework. The detailed numerical results are provided in table V.

C. Numerical Analysis and Discussion

The advantages of the proposed dual network solution is apparent from the numerical results reported in table V. The results of the baseline methods are produced from recently published [16] and we use the exact same train-validation-test partition on the datasets to ensure fairness. From the table V, we observe that proposed DNS consistently outperforms the baseline and the state-of-the-arts methods by significant margins (around 10% lesser MSE from the next best model). The RBF network (100 hidden nodes) with past dependency estimated performs second best among the competing methods, however, it cannot attain a good movement prediction (**MPM** $\approx 48-52\%$). On the other hand with DNS the same RBF network can provide a **MPM** $\approx 77-94\%$ as shown in table III and IV. This further indicates to an overall improved lag-free forecast attained by the same network when adopting DNS. The superior performance of DNS can be attributed to the followings,

- N_1 maps external features to the extracted trend (with MAM or SDM) which allows DNS to predict a sudden change in trend direction even when the consecutive past movements have been similar in nature thus **high MPM** is achieved (table III and IV).
- N_2 utilizes the predicted trend from N_1 to drive the forecast. Thereby the predicted curve follows the actual curve closely in a lag-less manner hence, DNS achieves **lower MSE** than its peers.
- PDE helps to reduce unwanted under or over-fitting (like utilization of less or more past instances than required can lead to those).
- DNS is advantageous in terms of computational cost as well. Compared to the resource-heavy deep networks (i.e. LSTM, CNN, CNN-LSTM), DNS only learns 200 weights (N_1+N_2) over 100 epochs.

Remark 6 : We run the experiments with DNS over 10 independent trials and provide the average MSE in table V

TABLE V: Performance comparison of DNS with baseline methods

Model	**Stock MSE**			**Power MSE**		
	1 step	**3 step**	**5 step**	**1 step**	**3 step**	**5 step**
Naive	5.62	17.46	28.78	0.42	1.22	1.67
ARIMA [1]	5.70	19.97	34.33	0.42	1.19	1.60
SVR [28]	5.58	17.68	28.69	0.42	1.15	1.55
ANN [29]	5.60	17.63	28.69	0.41	1.16	1.61
LSTM [4]	5.60	17.27	28.59	0.40	1.14	1.59
CNN [30]	5.65	17.21	28.29	0.42	1.14	1.54
State Frequency Memory [8]	5.57	17.00	28.90	-	-	-
Attentive Neural Network [16]	5.57	17.00	28.22	0.40	1.13	1.54
Ensemble CNN & LSTM [22]	5.61	17.32	28.46	0.41	1.13	1.56
RBF[23] with required past instances	5.27	16.54	27.45	0.33	0.92	1.52
Dual Network Solution (DNS)	**4.56**	**14.33**	**24.16**	**0.27**	**0.81**	**1.48**

to ensure reproducibility. The standard deviation between the runs is a nominal 5% of the mean MSE.

Conclusion

A regular time series model can detect a change in the trend only when the historical data has also gone through the same, hence the prediction is always late leading to the problem of lag. In this paper, we have presented a solution where external features are employed to predict the trend behavior. The predicted trend is then utilized to drive the forecast closely with the target sequence in a lag less manner. Thus the Dual Network Solution provides a better forecast both in terms of low MSE and better movement prediction (high MPM) when compared to other methodologies. In the future, we plan to build a self-evolving DNS to handle online time series forecasting.

References

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[2] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[6] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[7] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," in *Advances in Neural Information Processing Systems*, 2016, pp. 3882–3890.

[8] H. Hu and G.-J. Qi, "State-frequency memory recurrent neural networks," in *International Conference on Machine Learning*, 2017, pp. 1568–1577.

[9] R. Mittelman, "Time-series modeling with undecimated fully convolutional neural networks," *arXiv preprint arXiv:1508.00317*, 2015.

[10] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2014.

[11] S. Samanta, S. Suresh, J. Senthilnath, and N. Sundararajan, "A new neuro-fuzzy inference system with dynamic neurons (nfis-dn) for system identification and time series forecasting," *Applied Soft Computing*, vol. 82, p. 105567, 2019.

[12] S. Samanta, S. Ghosh, and S. Sundaram, "A meta-cognitive recurrent fuzzy inference system with memory neurons (mcrfis-mn) and its fast learning algorithm for time series forecasting," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 2231–2237.

[13] S. Samanta, A. Hartanto, M. Pratama, S. Sundaram, and N. Srikanth, "Rit2fis: A recurrent interval type 2 fuzzy inference system and its rule base estimation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[14] S. Samanta, M. Pratama, and S. Sundaram, "A novel spatio-temporal fuzzy inference system (spatfis) and its stability analysis," *Information Sciences*, vol. 505, pp. 84 – 99, 2019.

[15] K. Subramanian and S. Suresh, "A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system," *Applied soft computing*, vol. 12, no. 11, pp. 3603–3614, 2012.

[16] Y. Zhao, Y. Shen, Y. Zhu, and J. Yao, "Forecasting wavelet transformed time series with attentive neural networks," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1452–1457.

[17] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

[18] G. S. Atsalakis and K. P. Valavanis, "Forecasting stock market short-term trends using a neuro-fuzzy based methodology," *Expert systems with Applications*, vol. 36, no. 7, pp. 10 696–10 707, 2009.

[19] H.-x. Zhao and F. Magoulès, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, 2012.

[20] T. Alexandrov, S. Bianconcini, E. B. Dagum, P. Maass, and T. S. McElroy, "A review of some modern approaches to the problem of trend extraction," *Econometric Reviews*, vol. 31, no. 6, pp. 593–624, 2012.

[21] E. Gonzalez-Romera, M. A. Jaramillo-Moran, and D. Carmona-Fernandez, "Monthly electric energy demand forecasting based on trend extraction," *IEEE Transactions on power systems*, vol. 21, no. 4, pp. 1946–1953, 2006.

[22] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, no. CONF, 2017, pp. 2273–2279.

[23] M. J. Orr *et al.*, "Introduction to radial basis function networks," 1996.

[24] G. S. Babu and S. Suresh, "Meta-cognitive rbf network and its projection based learning algorithm for classification problems," *Applied Soft Computing*, vol. 13, no. 1, pp. 654–666, 2013.

[25] S. Samanta, M. Pratama, S. Sundaram, and N. Srikanth, "Learning elastic memory online for fast time series forecasting," *Neurocomputing*, 2019.

[26] A. Das, N. Anh, S. Suresh, and N. Srikanth, "An interval type-2 fuzzy inference system and its meta-cognitive learning algorithm," *Evolving Systems*, vol. 7, no. 2, pp. 95–105, 2016.

[27] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 2141–2149.

[28] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[29] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," *Expert Systems with Applications*, vol. 42, no. 2, pp. 855–863, 2015.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] L.-J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1506–1518, 2003.